ITU AI/ML in 5G Challenge

ITU-ML5G-PS-015: Network failure detection and root cause analysis in 5GC by NFV-based test environment.

Team：YOTA-YOTA
Members
Akiyoshi Shota ・Yamaguchi Yu・Oshima Saiga・Tashiro Ryoga

1. Introduction

This paper explores network fault detection and classification to use AI/ML within the 5G core network in NFV-based test environment.

2. Theme

Datasets are given, which have labels showing network fault type, time record of failure occurred and state data in the 5G core network. We classify these datasets by failure type using ML.

Table 1：name of datasets

Area	Urban	Middle	Rural
Dataset	Network-5gc-a	Network-5gc-b	Network-5gc-c
Label	Failure-label-a	Failure-label-b	Failure-label-c

2.1 Task 1

This task's goal is failure detection at Urban and Rural areas. Datasets acquired espectively at Urban and Rural areas are given.

2.2 Task 2

This task's goal is failure detection at Middle area between Urban area and Rural area. However, the amount of dataset at Middle area is quite smaller , compared with other areas(i.e., Urban and Rural). Thus, ML have high difficulty, compared with task 1.

3. Methods

3.1 The method of extracting attribute

We extracted attributes as follows:

- Confirming each failure types occurred and interface state.
- Exploring attribute transition at before and after failure

3.1.1 Confirming each failure types occurred and interface state.

Each failure and interfaces as cause of failure are shown in Table 2. Hereinafter, failure is represented as labels following Table 2.

Table2：Failure types and its number with each interfaces.

<table>
<tr><th>Failure types and Labels</th><th colspan="2">Interfaces</th></tr>
<tr><td>0. normal</td><td colspan="2"></td></tr>
<tr><td>1.bridge-delif(addif)</td><td colspan="2" rowspan="3">udm ens4
amf ens5
ausf ens4</td></tr>
<tr><td>2.interface-down(up)</td></tr>
<tr><td>3. interface-loss-start(stop)</td></tr>
<tr><td>4. memory-stress-start(stop)</td><td rowspan="2">udm
amf
ausf</td><td></td></tr>
<tr><td>5. vcpu-overload-start(stop)</td><td>Core0
Core1</td></tr>
</table>

3.1.2 Exploring attribute transition at before and after failure

In this section, we show a method for exploring parameter which is capable to be attribute. We explored attribute transition at before and after failure with focus on interfaces within each failure. Table 3 shows parameters as attribute we selected. The

features during the disruption did not change in all areas. We used these parameters all as attribute.

Table 3：Parameters as attribute we selected

Label	Parameter	Features in failure
1	in-octets	“0” streak is seen.
2	oper-status	Change “Up” to “Down”
3	out-octets	“0” streak is seen.
4	memory-status	Change “Healthy” to ”Critical”
	used-percent	The percentage changes around “80%” to around “90%”
5	user	Increase from less than 1% to about 25%.
	system	Increase from less than 1% to about 75%.
	idle	Decreased from values above 90% to about 0%.

3.2 Data pre-processing

In this section, we explain about data pre-processing which original data is processed into training data. Fig.1 shows the outline of data pre-processing.

First, we arranged labels which show network data and duration of failure in chronological order, and removed unstable data for 20 seconds caused by shift normal and failure state, by following instructions of questioner. The label is one set of failure occurrence and recovery. We deal with data between failure occurrence and recovery as failure data.

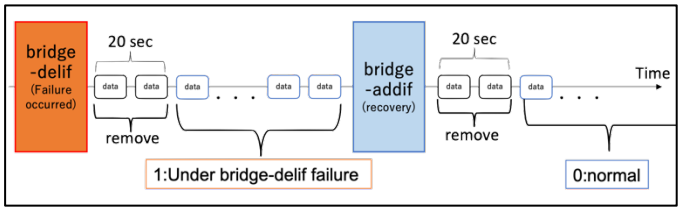

Fig.1：Image of data pre-processing

Next, We explain about the method of extraction attribute for training data from original data. We acquired all interfaces data under failure which is shown Table 3, and processed these data like a Table 4. We use 33 attributes for training.

Table 4：The way of processing parameters

Parameters	Way of processing	The number of interfaces subject to processing
in-octets	amount of change in 10 seconds	3
out-octets		3
oper-status	represent by 0 as “up” and 1 as “down”	3
memory-status	represent by 0 as “Healthy” and 1 as “Critical”	3
used-percent	represent by converting percentage value into ratio	3
user		6
system		6
idle		6

3.3 Data details pre-processed

In this section, we show data pre-processed by the method proposed by our team. The problem with processing the original JSON file as is that it is slow. As a solution, the data for training and testing was written from the original JSON file to a TXT file and read before training. We show results of data pre-processing as previously stated following bellow.

Table 5：Compare amount of data

	Original data	Post-processing data
Urban area	32[GB]	18[MB]
Rural area	32[GB]	18[MB]
Middle area	9[GB]	5[MB]
Total	73[GB]	41[MB]

Table 6：Details of training data

Label	Urban	Rural	middle
0	87649	87690	11000
1	1226	1230	152
2	1142	1119	146
3	1139	1141	145
4	1172	1157	149
5	1177	1178	148
Total	93505	93515	11740
Failre / Total	6.70%	6.60%	6.70%

Table7：Details of test data

Label	Urban	Rural	middle
0	21900	21914	21891
1	308	306	312
2	287	278	286
3	280	284	287
4	291	292	295
5	294	292	292
Total	23360	23366	23363
Failre / Total	6.70%	6.60%	6.70%

The original data is large, 73 [GB] in total, and it is difficult to train all of it. In order to make it possible to learn in a practical learning time with limited computing power, it is essential to reduce the amount of data. However, if the data is reduced in such a way that the features of each failure are lost, learning will not proceed properly. Therefore, it is necessary to reduce the amount of data without losing the features of each failure.

Table 5 shows the amount of data resulting from the necessary processing of the original data. Our proposed method can reduce the amount of data to 99.95% of the original data. In addition, the classification accuracy is nearly 97% in all tasks, which means that we can reduce the data without losing much of the features.

3.4 Learning model selection

In this section, we describe the learning model we used and the reasons why we chose it. We chose the learning model based on the following two points. (1) It should be good at classifying classes. (2) It should be able to learn the salient features of the dataset well. Within these conditions, we focused on XGBoost and LightGBM, which are capable of learning with such high accuracy that they occupy top positions in recent learning contests.

Table 8 shows the comparison of classification accuracy when using XGBoost and LightGBM in each task. As a result of the comparison, XGBoost showed higher classification accuracy in all tasks, so we decided to use XGBoost.

Table 8: Comparison of classification accuracy between XGBoost and LightGBM

	Algorithm	
Classified object	XGBoost	LightGBM
Urban	0.96609589	0.96091609
Rural	0.984165026	0.98219635
middle	0.972178231	0.96819757

In the comparison of the algorithms in this task, XGBoost outperformed LightGBM in all tasks. This can be attributed to the different learning strategies of the two algorithms. XGBoost branches the decision tree without prioritizing the leaves in Level-wise. LightGBM, on the other hand, learns in a leaf-wise manner, where leaves are prioritized for branching. In general, LightGBM is an improved version of XGBoost, and is capable of faster and more accurate learning. However, XGBoost may be better for some datasets, so comparison is necessary when using it.

3.5 How to generate the training data

In this chapter, we describe the issues and our approach to data usage in each task.

3.5.1 How to generate the training data for Task 1

In Problem 1, we have two types of data for fault detection: urban and rural. We considered two learning models, shared model (using data from urban and rural areas) and individual model (using data from urban or rural

areas). We decided to compare the classification accuracies of each model to determine which model to use since they can be trained using the same features in two different areas. Table 9 shows the classification accuracies of the share model and the individual models. As a result of the comparison, we decided to adopt the share model with the higher classification accuracy.

Table 9: Comparison of classification accuracy between share and individual models

	Share model		Individual model	
Training data	Urban , Rural		Urban	Rural
Test data	Urban	Rural	Urban	Rural
Accuracy	0.966	0.984	0.962	0.983

The accuracy of the share model was higher than that of the individual model for Task 1. This is due to the fact that there was no difference in the features of urban and rural areas and that the training data was more abundant. The numerical changes during disability in all urban, intermediate, and rural areas vary as shown in Table 3.Due to these factors, we believe that the common model was able to create a learning model with high generalization performance from abundant data.

Consider the differences in data between urban and rural areas. The difference between the areas could be the increase or decrease of each parameter according to the population. When the five classification targets are impaired, the values change to near zero or the status becomes impaired. In contrast, it is conceivable that the selected features increase or decrease according to the population when they are normal. There is no effect of area on learning using decision trees.

3.5.2 How to generate the training data for Task 2

In Task 2, the amount of training data is about 1/8 of that in Task 1. This makes it difficult to learn well and to solve the problem by increasing the number of training epochs. Therefore, we worked on increasing the amount of data. There were two ways to increase the amount of data: (1) a method to pseudo-increase the amount of data by learning, and (2) a method to borrow the data set of task 1. In the method (1), the features are likely to be different from the actual data, and the generalization performance is likely to be degraded. Therefore, we tried the method (2). Table 10 shows the classification accuracy of task 2 for different borrowed datasets. Table 10 shows the classification accuracy for different borrowed datasets. Based on these results, we decided to add the urban dataset from task 1, which has the highest classification accuracy, to our training data.

Table 10: Classification accuracy for different data sets

Training data	Middle	Middle Urban	Middle Rural	Middle Urban Rural
Accuracy	0.9694	0.9721	0.9690	0.9686

In Task 2, the model trained by adding the urban data from Task 1 to the middle part of the model was the most accurate. This was due to the fact that we were able to increase the amount of training data with the same trend as in task 1. Considering the amount of data alone, it would seem that the model trained using all the data would have the highest accuracy, but this was not the case this time. This is due to the fact that the rural data contains data that cannot acquire generalization performance for the data in the middle part. In fact, the results in Table 10 show that the inclusion of rural data is worse than the results for the middle part alone.

3.5.3 Summary of learning methods

Based on the results so far, we take the following learning approach for each task. We use XGBoost as the learning algorithm for all tasks. In task 1, we use the share model. In task 2, we add the urban area data from task 1 to the middle data as training data. Fig. 2 and Fig. 3 show an image of the learning model for each task.

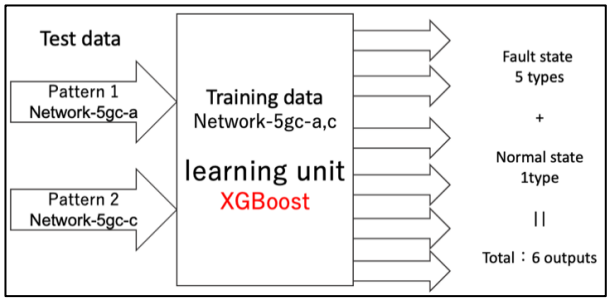

Figure 2: Learning model for Task 1

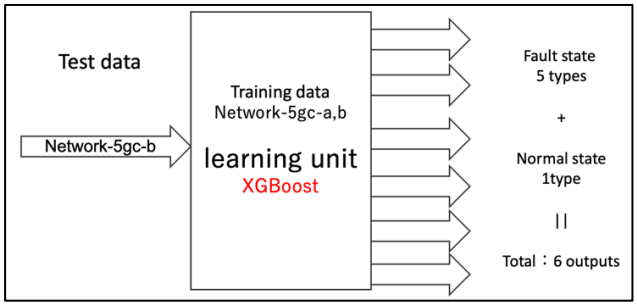

Figure 3: Learning model for Task 2

4. Results

4.1 Results for task 1 Urban Area

The total accuracy in the urban area of task 1 is 96.61%. Table 11 shows the Confusion matrix, Table 12 shows each elements of the accuracy, precision, recall, and F-measure, and Fig. 4 shows a graph of Table 12.

Table 11：Confusion matrix

predict ＼ True	0	1	2	3	4	5
0	21354	15	519	11	1	0
1	82	222	0	4	0	0
2	0	0	287	0	0	0
3	137	23	0	120	0	0
4	0	0	0	0	291	0
5	0	0	0	0	0	294

Table 12 : Accuracy, precision, recall, and F-measure

	0	1	2	3	4	5	Ave.
Accuracy	0.96609589						
Precision	0.99	0.854	0.356	0.889	0.997	1	0.848
Recall	0.975	0.721	1	0.429	1	1	0.854
F-measure	0.982	0.782	0.525	0.578	0.998	1	0.811

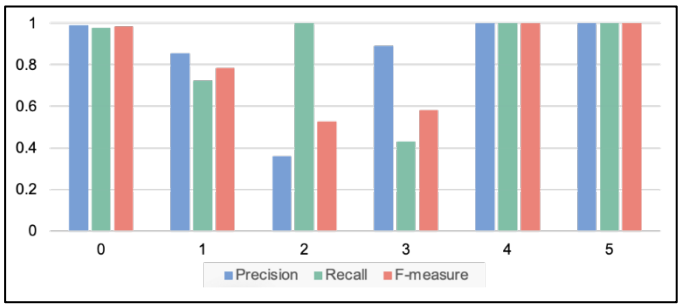

Figure 4 : Graph of Table 12

4.2 Results for Issue 1 Rural Areas

The accuracy in the rural area of task 1 is 98.42%. Table 13 shows the Confusion matrix, Table 14 shows the accuracy, precision, recall, and F-measure, and Fig. 5 shows a graph of Table 14.

Table 13：Confusion matrix

predict ＼ True	0	1	2	3	4	5
0	21884	9	0	21	0	0
1	121	180	0	5	0	0
2	0	0	278	0	0	0
3	209	5	0	70	0	0
4	0	0	0	0	292	0
5	0	0	0	0	0	292

Table 14 : Accuracy, precision, recall, and F-measure

	0	1	2	3	4	5	Ave.
Accuracy	0.984165026						
Precision	0.985	0.928	1	0.729	1	1	0.94
Recall	0.999	0.588	1	0.246	1	1	0.806
F-measure	0.992	0.72	1	0.368	1	1	0.847

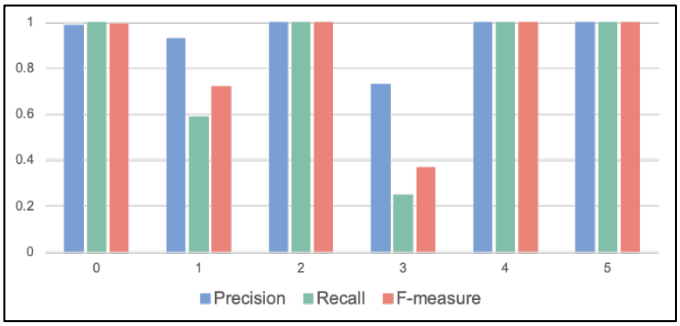

Figure 5 : Graph of Table 14

4.3 Results for the middle of Task 2

The total accuracy in the middle of task 2 is 97.22%. Table 15 shows each elements of the Confusion matrix, Table 16 shows the accuracy, precision, recall, and F-measure, and Fig. 6 shows a graph of Table 16.

Table 15：Confusion matrix

predict ＼ True	0	1	2	3	4	5
0	21849	6	0	32	4	0
1	270	38	0	4	0	0
2	95	0	191	0	0	0
3	235	4	0	48	0	0
4	0	0	0	0	295	0
5	0	0	0	0	0	292

Table 16 : Accuracy, precision, recall, and F-measure

	0	1	2	3	4	5	Ave.
Accuracy	0.972178231						
Precision	0.973	0.792	1	0.571	0.987	1	0.887
Recall	0.998	0.122	0.668	0.167	1	1	0.659
F-measure	0.986	0.211	0.801	0.259	0.993	1	0.708

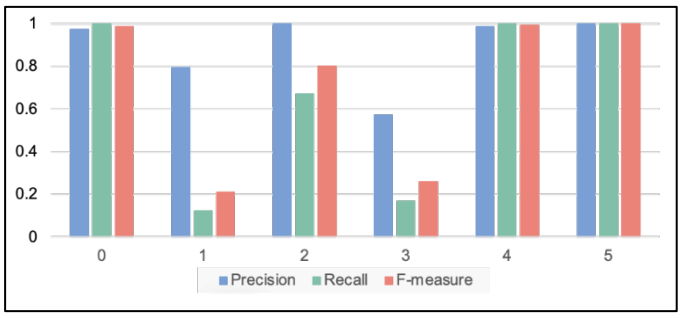

Figure 6 : Graph of Table 16

5. Discussion of the results

From here, the results are discussed, and finally, improvements to the proposed method are discussed.

5.1 Discussion of the results

In this section, we consider each failure.

Fault 5 was found to be completely detectable by the method used in this study.

Fault 4 is not completely detected, but it can be classified with high accuracy. Fault 4 can be classified with high accuracy, though not completely. However, all of these areas had zero false positives and zero omissions, so we can say that they are practical.

Obstacles 1 and 3 had a low percentage of correct answers in all areas. This is because only the amount of transmission and reception was given as the feature value. In the case of failure 1 and 3, the correct answer rate was low in all areas. However, the value of 0 appears periodically even in normal conditions. If we look at the time series, we can determine that the value of 0 is continuous in the case of failure. Since our algorithm cannot determine the time series, it is difficult to make a decision based on this feature alone. Therefore, we believe that it was not possible to distinguish between normal and abnormal 0s.

As for obstacle 2, we were able to classify the rural areas completely, but the accuracy was not good in the urban and middle areas. Looking at the urban areas first, there were 287 correct answers that matched the labels and predictions, and 519 false positives. The breakdown of the data shows that 287 of them were for fault 2, and all of them were correctly predicted. The 519 false positives may be due to the oper-status parameter being set to "down" even though the data is normal, and a new feature is needed to distinguish these. Next, when we look at the middle section, 191 out of 286 data, which are all failures 2, can be correctly classified, and 95 are not classified as normal. This indicates that there may be some data in the middle section whose oper-status parameter is still Up at the time of failure. A new feature is needed to distinguish these as well.

5.2 Future improvements

Consider the operation of fault detection in a real environment. In a real environment, it is more important to avoid false positives than false negatives. Therefore, it is necessary to improve the reproduction rate of failures 1, 2, and 3 by adding new features to eliminate the omissions.

6. Summary

In this study, we aimed to detect anomalies and determine the type of anomaly using data measured in various parts of an NFV that simulates a 5G core network. Urban and rural fault detection was done in task 1 and middle area fault detection was done in task 2. We extracted 33 features from the changes of parameters before and after the failure of the data set and used them for training.

We took the following approach：

- Feature extraction from original data
  - Achieved 99.95% reduction in data volume
- Verification and selection of the best algorithm
- Task 1: Improve accuracy by using a common model
- Task 2: Increasing the amount of training data using the data from Task 1

We used XGBoost for failure detection and classification.

As a result, the accuracy was 99.61% in the urban area of task 1, 98.42% in the rural area, and 97.22% in the middle area of task 2. We were able to classify failures 4 and 5, memory-stress and vcpu-overload, with particularly high accuracy, demonstrating the effectiveness of our proposed method.

As a future improvement, it is necessary to make improvements to reduce the number of failures by adding more features.